Package 'DDHFm'

October 25, 2016

Title Variance Stabilization by Data-Driven Haar-Fisz (for Microarrays)

Version 1.1.2

Date 2016-10-20

Description Contains the normalizing and variance stabilizing Data-Driven Haar-Fisz algorithm. Also contains related algorithms for simulating from certain microarray gene intensity models and evaluation of certain transformations. Contains cDNA and shipping credit flow data.

Depends R (>= 2.10)

Imports lokern, wavethresh

License GPL-2

NeedsCompilation yes

Author Efthimios Motakis [aut],
Guy Nason [aut, cre],
Piotr Fryzlewicz [aut]

Maintainer Guy Nason <g.p.nason@bristol.ac.uk>

Repository CRAN

Date/Publication 2016-10-25 00:16:10

R topics documented:

---

cdna   *Example cDNA data*

Description

cDNA data from a one-colour experiment from the Stanford Microarray Database. The data set is stored as an intensity matrix with 42624 genes (rows) and 4 replicates (columns).

The experiment numbers are 40430, 40571, 34905 and 34912.

Usage

```
data(cdna)
```

Format

A 42624x4 matrix

Source

Stanford Microarray Database

References

McCaffrey, R.L., Fawcett, P., O'Riordan, M. Lee, K., Havell, E.A. Brown, P.O. and Portnoy, D.A. (2004) A specific gene expression program triggered by Gram-positive bacteria in the cytocol, Proc. Nat. Acad. of Science, 101, 11386-11391

---

cDNAdata   *Samples from the cDNA data vector*

Description

Arranges a vector of intensities into a form amenable to analysis by the DDHF and also can restrict the number of genes analysed and also obtain a random sample

Usage

```
cDNAdata(data.vect,cdnalength,datasize,ng,nrep)
```

Arguments

data.vect	The data vector of intensities
cdnalength	Only considers the first cdnalength genes for analysis
datasize	Needs to be a power of two. The number of genes that get randomly sampled from the first cdnalengthq and then subsequently transformed.
ng	The total number of genes described by data.vect
nrep	The number of replicates (should be a power of two) contained in data.vect

Details

The

$$J = ng \times nrep$$

vector data.vect should contain first the intensities of the first replicate of the ng genes', then the second replicate of all of the ng genes in the same order, and so on.

This function first puts the one dimensional data.vect into a matrix form with ng rows and nrep columns (so the row number indices the gene and the column number the replicate).

Then the first cdnalength rows are extracted and kept, the remaining rows are discarded.

Then datasize rows at random are extracted and kept and the remaining rows are discarded.

Value

cDNAdata	The data vector in the proper format to perform Data-Driven Haar-Fisz algorithm

Author(s)

Efthimios Motakis <e.motakis@bris.ac.uk>

DDHFm   *Data-driven Haar-Fisz for microarrays*

Description

Takes a matrix containing the resuls of a (e.g.) microarray experiment (rows are different genes, columns are replicates) and performs a multiscale variance stabilization and Gaussianization transform. Note, you have to have replicates to make this work.

Usage

```
DDHFm(x)
```

Arguments

x   A matrix containing n rows (genes) and k columns (replicates)

Details

The data-drive Haar-Fisz transform is a multiscale variance stabilization and Gaussianization transform. The algorithm will run on a one-column matrix but the results will NOT be good. For one column of data use some other variance stabilizer (in which case why are you using one column, eh?)

However, if you have replicates then you can use this powerful multiscale algorithm which will stabilize variance and also draw the distribution of the intensities towards Gaussian extremely effectively (so skew and kurtosis will look good too).

This function forms a single vector of intensities by the following procedure. First, it computes the mean over replicates for each gene (this is why you need replicates, really). Then it orders the rows of the matrix into increasing mean order. Then it converts that matrix into a vector (which is therefore a sequence of intensities which are grouped according to gene (according to increasing mean intensity within each group)). Then this vector is padded up to the next power-of-two in length with zeroes. Then the padded vector is subjected to the ddhft.np.2 function which actually does the data-driven Haar-Fisz transform. The results of this are unpadded and then rearranged into a matrix with an identical structure to the input matrix. This final matrix contains stabilized and Gaussianized (is this a word?) intensities.

It is important that you have replicates (ie, the number of columns of the matrix has to be greater than 1). Also, it is scientifically important that the replicates be just that (ie, although the rows might refer to the same genes the columns must be results from repeats experiments held under identical conditions). For example, you shouldn't have results for the gene on one subject and then the same gene on another subject unless I suppose you are considering them to be replicates in some way.

Value

A single matrix with the same dimensions as the input matrix. Each entry in the resultant matrix corresponds to a variance stabilized and Gaussianized version of the entry in the input matrix.

Author(s)

Guy Nason

References

Motakis, E.S., Nason, G.P., Fryzlewicz, P. and Rutter, G.A. (2005) Variance stabilization and normalization for one-color microarray data using a data-driven multiscale approach. *Technical Report*, 05:16, Statistics Group, Department of Mathematics, University of Bristol, UK

Motakis, E.S., Nason, G.P., Fryzlewicz, P. and Rutter, G.A. (2006) Variance stabilization and normalization for one-color microarray data using a data-driven multiscale approach. *Bioinformatics*, **22**, 2547–2553.

See Also

ddhft.np.2

Examples

```
#
# First, let's make up some "true" intensities (these are gamma)
#
TrueInt <- genesimulator(nreps=5, nps=100)
#
# Now let us simulate data from the Durbin and Rocke (2001) model conditioning
# on the TrueInt
#
MAInts <- simdurbin2(TrueInt,alpha=24800, seta=0.227, seps=4800)
#
# Now put these intensities into the correct format for DDHFm. That is
# replicates across columns
#
MAm <- matrix(MAInts, ncol=5, byrow=TRUE)
#
# Apply DDHFm transform
#
MAmDDHFm <- DDHFm(MAm)
#
# Now look at the variances of each
#
diag(var(MAm))
#
#[1] 22443437 25065729 23430160 22470058 21023183
#
diag(var(MAm))/ mean(diag(var(MAm)))
#
#[1] 0.9806403 1.0952183 1.0237540 0.9818035 0.9185839
#
# Now for the DDHFm version
#
diag(var(MAmDDHFm))
#
```

```
# [1] 10938.71 11369.41 11169.50 11144.74 10959.94
#
diag(var(MAmDDHFm))/ mean(diag(var(MAmDDHFm)))
#
# [1] 0.9840099 1.0227549 1.0047711 1.0025441 0.9859200
#
# So stabilization is better with DDHFm
#
```

---

ddhft.np.2   *Data-Driven Haar-Fisz transformation*

Description

Forward Data-Driven Haar-Fisz transform

Usage

```
ddhft.np.2(data, Ccode=TRUE)
```

Arguments

data	A vector of size 2^J containing the data to variance stabilize and Gaussianize.
Ccode	If TRUE then fast C code is used, otherwise R code is used

Details

Performs the data-driven Haar-Fisz transform on sequence data. This consists of (i) the Haar wavelet transform of sequence; (ii) estimation of mean-variance relationship between finest level smoothing and detail wavelet coefficients using isotonic regression (see `isotone`); (iii) divide wavelet detail coefficients by smooth ones subjected to the estimated mean-variance relationship; (iv) perform the inverse Haar wavelet transform of the modified coefficients.

The aim is to variance stabilize and Gaussianize the sequence data which is only assumed to be positive and possess an underlying increasing mean-variance relationship.

Value

A list containing the following components

hft	The data-driven Haar-Fisz transform of the input sequence
mu	The mu's obtained from the input sequence used for estimating the mean-variance relationship
sigma	The estimated standard deviation as a function of the mean, the result of the isotonic regression fit of sigma2 on sigma
sigma2	The local multiscale standard deviations associated with each mean
factors	The numbers that divide the detail coefficients to standardize variance (obtained from the mean-variance estimation)

Author(s)

Piotr Fryzlewicz <p.fryzlewicz@imperial.ac.uk>

References

Fisz, M. (1955), The limiting distribution of a function of two independent random variables and its statistical application, Colloquium Mathematicum, 3, 138-146.

Delouille, V., Fryzlewicz, P. and Nason, G.P. (2005), A data-driven Haar-Fisz transformation for multiscale variance stabilization. Technical Report, 05:06, Statistics Group, Department of Mathematics, University of Bristol

See Also

ddhft.np.inv

Examples

```
#
# Generate example Poisson data set.
#
# Intensity function is steps from 1 to 32 in steps of 4 with each intensity
# lasting for 128 observations. Then sample Poisson with these intensities
#
v <- rpois(1024, lambda=rep(seq(from=1, to=32, by=4), rep(1024/8,8)))
#
# Let's take a look at this
#
## Not run: ts.plot(v)
#
# Ok. So mean of intensity clear increasing, but variance increasing too
#
# Now do data-driven Haar-Fisz
#
vhft <- ddhft.np.2(v)
#
# Now plot the variance stabilized series
#
## Not run: ts.plot(vhft$hft)
#
# The variance of the observations is much closer to 1. For example, let's
# look a the variance of the original series and the transformed one
#
# For the first intensity of 1
#
var(v[1:128])
#[1] 0.6628322
#
var(vhft$hft[1:128])
#[1] 1.025151
#
#
```

```
# And for second intensity of 5
#
#
var(v[129:256])
#[1] 4.389518
var(vhft$hft[129:256])
#[1] 1.312953
#
# So both transformed variances near to 1
#
# Now plot the estimated variance-mean relationship
#
## Not run: plot(vhft$mu, vhft$sigma)
## Not run: lines(vhft$mu, sqrt(vhft$mu))
#
# This is an approximately square root function (because you expect the
# sd of Poisson to be the square root of the mean).
#
```

---

ddhft.np.inv   *Inverse Data-Driven Haar-Fisz transformation*

Description

Inverts the Data-Driven Haar-Fisz transform to obtain the raw (untransformed) data

Usage

```
ddhft.np.inv(hft.obj)
```

Arguments

hft.obj   An object with the same structure as that returned by the ddhft.np.2 function.

Details

Merely performs the inverse of the DDHF transform see ddhft.np.2

Value

The inverted transform

Author(s)

Piotr Fryzlewicz <p.fryzlewicz@imperial.ac.uk>

References

Delouille, V., Fryzlewicz, P. and Nason, G.P. (2005), A data-driven Haar-Fisz transformation for multiscale variance stabilization. Technical Report 05:06, Statistics Group, Department of Mathematics, University of Bristol.

See Also

ddhft.np.2

---

dhhrcomp	*Simulated genes, apply DDHFm then compute and return variance, skewness and kurtosis values*

Description

An example gene intensity simulator with given mean values mu0 which then returns the variance, skew and kurtosis of the DDHF transformed intensities. The gene intensity simulation is performed using simdurbin2

Usage

```
dhhrcomp(nsims = 1024, nmu = 4,
mu0 = c(0, 5, 10, 15, 20, 25, 30, 40, 50, 60,
65, 70, 80, 100, 200, 300, 500, 1000) * 1000)
```

Arguments

nsims	Number of replicates for each mu
nmu	Selects first nmu genes from mu for use
mu0	The possible gene intensity means

Details

This function simulates some gene intensities. A list of possible intensity means are supplied in mu0. The first nmu of these are selected. Then for each of the selected means nsims gene intensities are generated.

The intensities are them subjected to DDHF transformation.

Value

mu	A vector of length nmu of the mean intensities considered
v.hft	A vector of nmu variances, one for each mean in mu0. The variance is the variance of the DDHFm transformed simulated intensity data.
s.hft	A vector of nmu skewnesses, one for each mean in mu0. The skewness is the skewness of the DDHFm transformed simulated intensity data.
k.hft	A vector of nmu kurtoses, one for each mean in mu0. The kurtosis is the kurtosis of the DDHFm transformed simulated intensity data.

Author(s)

Guy Nason <g.p.nason@bris.ac.uk.

References

Durbin, B.P., Hardin, J.S., Hawkins, D.M. and Rocke, D.M. (2002), A variance-stabilizing transformation for gene expression microarray data, Bioinformatics, 18, S105-S110

See Also

[simdurbin2](#)

---

dhhrss	*Tabulates variance, skewness and kurtosis coefficients from the output of dhhrcomp*

Description

Tabulates the variance, skewness and kurtosis coefficients from the output of [dhhrcomp](#)

Usage

```
dhhrss(dhhrobj)
```

Arguments

dhhrobj   The results to be tabulated. The output of a call to [dhhrcomp](#)

Details

This routine merely runs summary statistics calculations on the output results from [dhhrcomp](#)

Value

None, prints a table.

Author(s)

Guy Nason <g.p.nason@bris.ac.uk.

---

function.from.vector	*Function applied for the computation of the Data-Driven Haar-Fisz transform*

Description

Applies the function which.min.diff to the argument.vect vector with the argument x

Usage

```
function.from.vector(x, y, argument.vect)
```

Arguments

x	a vector of data in ascending order
y	a vector with length(y) = length(x)
argument.vect	can be a vector

Value

function.from.vector

help function for data transformation

Author(s)

Piotr Fryzlewicz <p.fryzlewicz@imperial.ac.uk>

---

genesimulator	*Gene means simulator*

Description

Simulates means of the gene intensities for "nps" genes, each replicated "nreps" times. The mean is drawn from a gamma distribution with shape parameter "shape" and scale parameter "scale"

Usage

```
genesimulator(nreps = 3, nps = 100, shape = 4, scale = 100)
```

Arguments

nreps	Number of replicates
nps	Number of genes
shape	Shape parameter
scale	Scale parameter

Details

For many problems a set of reasonable gene mean intensities is useful for testing algorithms that later draw actual gene intensities with distributions that possess a gene mean intensity.

Value

A matrix containing nreps*nps rows and 3 columns. The first col contains all the gene mean intensities. The second and third col contain the gene replicate number and gene number respectively.

Author(s)

Guy Nason <g.p.nason@bris.ac.uk>

---

hftrialdatagen   *Gene intensities simulator and DDHFm tester*

Description

Simulates gene intensities and also applies DDHFm to them

Usage

```
hftrialdatagen(nreps = 4, nps = 128, plot.it = FALSE, uvp = 0.8)
```

Arguments

nreps	Number of replicates
nps	Number of genes
plot.it	Takes TRUE to activate the command of the respective plot and FALSE to deactivate it
uvp	a parameter for the denoising

Details

The code is well commented for further information.

First, [genesimulator](#) is called to obtain a vector of mean gene intensities (for a number of genes and a number of replicates for each gene.

Then link{simdurbin2} simulates a series of gene intensities using the (log-normal type) model as described in Durbin and Rocke (2001,2002).

Then for each gene the mean of replicates for that gene is computed.

Optionally, if plot.it is TRUE then the mean is plotted against its standard deviation (over replicates).

Then the intensities are sorted according to increasing replicate mean.

Optionally, if `plot.it` is TRUE then a plot of the intensities values as a vector (sorted according to increasing replicate mean) is plotted in black, and then the true mean plotted in colour 2 (on my screen this is red) and the computed replicate mean plotted in green.

The DDHF transform of the sorted intensities is computed.

Optionally, if `plot.it` is TRUE then a plot of the transformed means versus the transformed standard deviations is plotted. Followed by a time series plot of the transformed sorted intensities. These can be studied to see how well DDHF has done the transformation.

Then two smoothing methods are applied the the DDHF transformed data. One method is translation invariant, Haar wavelet universal thresholding. The other method is the classical smoothing spline. If `plot.it` is TRUE then these smoothed estimates are plotted in different colours.

Then the mean estimated intensity for each gene is computed and this is returned as the first column of a two-column matrix (`ansm`). The second column is the true underlying mean. The object `hftssq` contains a measure of error between the estimated and true gene means.

Value

`ansm`	Two column matrix containing the estimated gene intensities and the true ones
`hftssq`	Sum of squares between estimated means and true means
`yhf`	Simulated gene intensities

Author(s)

Guy Nason <g.p.nason@bris.ac.uk>

Examples

```
#
# First run hftrialdatagen
#
## Not run: v <- hftrialdatagen()
#
# Now plot the Haar-Fisz transformed intensities.
#
## Not run: ts.plot(v$yhf)
#
# Now plot the denoised intensities
#
# Note that above we have 128 genes and 4 replicates and so there are
# 4*128 = 512 intensities to plot.
#
# However, there are only 128 gene intensities, and estimates. So, for this
# plot we choose to plot the noisy intensities and then for each replicate
# group (which are colocated on the plot) plot the (necessarily constant)
# true and estimated intensities (ie we plot each true/estimated intensity
# 4 times, once for each replicate).
#
# First estimates...
#
## Not run: lines(1:512, rep(v$ansm[,1], rep(4,128)), col=2)
```

```
#
# Now plot the truth
#
## Not run: lines(1:512, rep(v$ansm[,2], rep(4,128)), col=3)
```

isotone	*Performs Isotone regression using "pool-adjacent-violators" algorithm*

Description

It is applied in order to estimate the h() function in the Data-Driven Haar-Fisz algorithm

Usage

```
isotone(x, wt = rep(1, length(x)), increasing = FALSE, Ccode=TRUE)
```

Arguments

x	the vector that will be fitted with the regression
wt	a vector of weights
increasing	if TRUE the curve is set to be increasing, else FALSE
Ccode	if TRUE then faster C code is used rather than R

Value

isotone	the regression results

Author(s)

Bernard Silverman, with C modifications by GPN

References

Johnstone, I.M. and Silverman, B.W. (2005), EbayesThresh: R and S-Plus programs for empirical Bayes thresholding, Journal of Statistical Software, to appear

---

| KURTOSIS | *Kurtosis Coefficient estimator* |

Description

Estimates the kurtosis coefficient of a data vector

Usage

```
KURTOSIS(x)
```

Arguments

x   the data vector

Value

The estimated kurtosis of x

Author(s)

Efthimios Motakis <e.motakis@bris.ac.uk

| MOMENTS | *Moment estimator* |

Description

Computes the rth central moment of the data x

Usage

```
MOMENTS(x, r)
```

Arguments

x   the data vector

r   parameter that describes which moment will be produced

Value

The rth central moment of x

Author(s)

Efthimios Motakis <e.motakis@bris.ac.uk.

ShipCreditFlow   *Example Shipping credit flow data*

Description

Shipping credit flow data. A set of monthly credit transaction flow series recorded by the Central Bank of Cyprus. The time range of the data is January 2002 to December 2014 and contains 156 observations.

Usage

```
data(ShipCreditFlow)
```

Format

A vector of length 156

Source

Central Bank of Cyprus

References

Michis, A. A. and Nason, G.P. (2016) Case Study: Shipping trend estimation and prediction via multiscale variance stabilisation. *Journal of Applied Statistics* (to appear)

---

simdurbin   *Gene intensities simulator*

Description

Simulates gene intensities from the two components model as in the papers of Durbin and Rocke.

Usage

```
simdurbin(n, alpha, mu, seta, seps)
```

Arguments

n	number of intensities to be simulated
alpha	background mean of gene intensities
mu	the means where the gene intensities are simulated from
seta	standard deviation of high-level gene intensities
seps	standard deviation of low-level gene intensities

Details

This function generates n intensities from the Durbin and Rocke (2001) gene intensity model with parameters alpha, seta and seps.

Value

A vector of the appropriate intensities.

Author(s)

Guy Nason <g.p.nason@bris.ac.uk>

References

Rocke, D.M. and Durbin, B.P. (2001), A model for measurement error for gene expression arrays, Journal of Computational Biology, 8, 557-569

See Also

[simdurbin2](#)

---

simdurbin2 *Gene intensities simulator*

Description

Simulates gene intensities from the two components model of Durbin and Rocke.

Usage

```
simdurbin2(mu, alpha, seta, seps)
```

Arguments

mu	the means where the gene intensities are simulated from
alpha	background mean of gene intensities
seta	standard deviation of high-level gene intensities
seps	standard deviation of low-level gene intensities

Value

Like [simdurbin](#) this function simulates gene intensities from the Durbin and Rocke two component model (see Durbin and Rocke, 2001). Unlike [simdurbin](#) this function generates each intensity with a mean specified by an entry in mu. So, length(mu) intensities are generated with intensity i having mean mu[i].

Author(s)

Guy Nason <g.p.nason@bris.ac.uk>

References

Rocke, D.M. and Durbin, B.P. (2001), A model for measurement error for gene expression arrays, Journal of Computational Biology, 8, 557-569

See Also

[simdurbin](#)

Examples

```
#
# Generate data with mu equal to 80 and mu equal to 80000 with 10 replicates each
#
#
v <- simdurbin2( c(rep(80, 10), rep(80000,10)), alpha=24800, seta=0.227, seps=4800)
#
# Let's look at the mean of the first 10 and the last 10
#
mean(v[1:10])
#[1] 27361.95
#
mean(v[11:20])
#[1] 92455.66
```

SKEW	*Skewness coefficient estimator*

Description

Estimates the skewness coefficient of the data

Usage

```
SKEW(x)
```

Arguments

x   The data vector

Value

The skewness of x

Author(s)

Efthimios Motakis <e.motakis@bris.ac.uk>

---

which.min.diff *Find index where two vectors are closest*

Description

Finds the index point where two index vectors are closest in value

Usage

```
which.min.diff(a, vect)
```

Arguments

a   input vector

vect   input vector

Value

The index where abs(a-vect) is smallest, ie where a and vect are smallest.

Author(s)

Piotr Fryzlewicz <p.fryzlewicz@imperial.ac.uk>

Examples

```
#
# Make up two vectors
#
a <- c(1,2,3)
vect <- c(3,2,1)
#
# Now see on which index are the two closest in value
#
which.min.diff(a,vect)
#[1] 2
#
# ie its the second index where both vectors are actually 2
```

Index